Los Alamos NATIONAL LABORATORY
research note
Applied Theoretical & Computational Physics Div.
X-TM:Transport Methods Group

To/MS: Distribution
From/MS: Todd Urbatsch, XTM MS D409
Tom Evans, XTM MS D409
Phone/FAX: (505)667–3513/(505)665–3677

Symbol: XTM-RN(U)-99-018 (Rev. 5)(LA-UR-99-3482)
Date: June 28, 1999

Subject: Software Testing in Milagro, Draco, and the Jayenne Project (Revised)

Executive Summary

Milagro is an Implicit Monte Carlo (IMC) code for XYZ, thermal radiative transfer. Draco is a library of components reused in various transport applications. Jayenne is the umbrella project encompassing our entire set of IMC efforts. To verify and maintain code stability and correctness, we have developed a three-level regression test program. Level 1 contains incrementally advanced component tests of C++ classes used in Milagro. Level 2 contains test problems that are run at the Milagro code level. Level 3 contains verification problems to test the correctness of the physics output from Milagro. Level 1 and 2 tests are executed nightly. Level 3 tests are executed weekly. All tests are automated under the Draco and Milagro build systems using python scripts.

1. Revision History

[Revision 1 (October 21, 1999): Contains descriptions of new nightly regression tests problems that test the user-defined surface source cells capability. Also contains corrections to the descriptions of nightly regression tests str01 through str06.]

[Revision 2 (November 12, 1999): Contains descriptions of new regression tests for the constant external material volume source. Also contains an addition of the Marshak-2A problem in fulltp06.]

[Revision 3 (March 17, 2000): Added three new regression tests: one with all surface source, one with all volume emission, and one with all census. Then, all three are restarted from the first, second, or third cycle and run out. Filenames are start?? and restart?? (01, 02, 03). They run both serially and in parallel.]

[Revision 4 (July 12, 2000): Added three new sets of parallel regression tests: start/restart?? (04, 05, 06) that are equivalent to the first three except using a full domain decomposition (DD) topology. Also added problems "restart07" through "restart12," which test restarting with alternate topologies (full DD and full replication).]

[Revision 5 (September, 2000): Added a small discussion on the RZ-Wedge Mesh regression tests, and added tables describing the tests.]

[Revision 6 (5 April 2001): Added OS_Mesh test problems for the modified Marshak-2B problem (specific heat increased from 0.1 to 1.0): fulltp07 and tp07.]

[Revision 7 (31 July 2001): Added problems inf31–inf34, b_start01, b_restart01–02, and rzinf15 to test the new input options for opacity: the "tlinear" (κ_1/T) and the "analytic_opacity_offsets:".]

[Revision 8 (12 December 2001): Removed testing of the Milagro–specific opacity and equation-of-state input because it is now handled with the Common Data Interface (CDI). No more "tlinear," "tcube," and "ana-

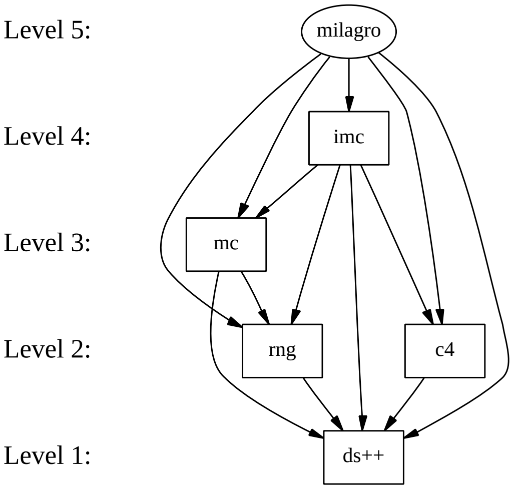

FIG. 1: Levelized component design of Milagro. Components labeled with boxes are from Draco. Components in ovals are part of Milagro proper.

lytic_opacity_offsets:" in the Milagro input. Removed inf18, inf21-24, and inf31-33, and then renumbered.]

[Revision 9 (13 February 2002): Added a suite of tests for Milagro's multigroup frequency treatment.]

2. **Overview**

Milagro [1] is an object-oriented code that is built from many component packages. For example, Milagro utilizes the RNG package for random number generation and the C4 package for message passing. A significant number of these packages reside in the Draco [2] code library. The remaining packages reside within the Milagro source directory tree. Each package utilized by Milagro is implemented according to the precepts of levelized design [3] as shown in Fig. 1. Additionally, each component package contains a levelized tree of classes that make up the component. Thus, one level of regression testing is to incrementally test the components and classes that are used to build Milagro.

The end product of Milagro is a stand-alone IMC code. This executable is the dynamic interaction of the components illustrated in Fig. 1. Additionally, the purpose of Milagro is to produce "correct" output from a given set of initial conditions. Hence, two additional levels of regression testing present themselves. First, given a set of initial conditions Milagro should produce consistent output from the synthesis of its individual components. Second, the output should be verifiable. In other words, if components are changed only the output that depends on those components should change. Also, given a physical description of a problem Milagro should produce a "correct" solution within the limits imposed by the IMC method.

In summary, we have developed three levels of regression testing in Milagro. The first level checks individual components for correctness. They should perform their designed services correctly. Because Milagro is a levelized design, this testing can be built up incrementally [3]. The second level of regression testing checks the dynamic interaction of all code components at the executable level. Given a set of initial conditions the code should produce predictable output that is constant over time. When changes are made in certain components, affected output should be dependent upon the code that has been modified. This level of testing ensures that unpredicted side-effects do not appear in the code after modifications to certain components.

The final level of regression testing is code verification. In the second level of regression testing, problems are run to test the stability and interaction of Milagro code components. In this level of regression testing,

TABLE 1: Summary of regression test levels in Milagro.

Regression Test Level	Schedule	Description
1	nightly	Draco component tests
2	nightly	Milagro test problems
3	weekly	Milagro verification problems

real physics problems are executed. The output from these executions can be used to verify that the code continues to produce "correct" physics output with the edition of new features.

The execution schedule for regression tests is dependent upon the level. Draco components are tested nightly as part of the Draco regression test suite. Milagro level 2 regression tests are performed nightly. On a sequential schedule, one of the Milagro level 3 verification problems are performed each weekend. The lower frequency of the level 3 tests is dictated by their lengthy runtimes of up to 48 CPU hours. Table 1 summarizes the three levels of regression testing in Milagro. The execution of all regression tests has been updated from the **dejagnu** [4] testing framework to a much less complicated python script.

The remainder of this note will detail the tests that comprise the three levels of regression testing in Milagro. Section 3 describes the component tests that form the first level of testing. Section 4 describes the level 2 test problems. Section 5 details the level 3 verification problems. The tests described in this note have been designed for the following releases of Milagro components:

Milagro	2_2_0
IMC	2_2_0b
MC	2_2_0
VIZ	1_1_0
RNG	1_4_0
C4	1_5_0
TRAITS	1_3_0
DS++	1_4_0

As new revisions of these components are released, the regression test suite will be updated and documented accordingly.

3. Level 1: Component Tests

Each Milagro component is tested nightly to verify code correctness. In this section we will expound the tests for the RNG, MC, and IMC components. The DS++ and C4 components are more general Draco packages and are more properly described elsewhere. Also, Milagro itself presently does not contain any component tests. As new components are added to the Milagro source tree, tests will be manufactured. At the time of this writing, the Milagro source tree consists only of instantiations of Draco components and a C++ `main` program.

The purpose of component tests is to verify the correctness of individual classes. In other words, given correct arguments class members should perform the proper operations. We use Design-By-Contract [5] to perform part of this task. The other part is performed by component tests. Essentially, these tests instantiate objects of a certain class and execute the various member functions that the class provides. The results of the function calls can be compared to *a priori* known output. When discrepancies occur failure messages are emitted. These messages are tallied and reported by the component test python script. If no failures are

TABLE 2: Component tests in the Draco RNG package.

Problem	Description
tstSprng	tests the Sprng random number class; tests random number generation, copying, and spawning

TABLE 3: Component tests in the Draco MC package.

Problem	Description
tstCoord	tests the Coord_sys, XYCoord_sys, and XYZCoord_sys classes; tests inheritance, position sampling and dimensionality services
tstOSMesh	tests the OS_Builder, Layout, and OS_Mesh classes; given a shunt interface the builder creates a mesh of known dimensions and then tests OS_Mesh functions including equality, dimensionality, normal calculations, connectivity, surface calculations, and distance-to-boundary evaluation

tallied then the script reports that the test passed.

These tests work in conjunction with the level 2 tests. For example, a level 2 test failure could indicate problems in any number of classes. The hope is that a level 2 failure would correspond to an appropriate level 1 failure that pinpoints the location of the error.

3.1. Component Tests

Levelized design makes component testing possible and relavent. Any component may assuredly used tested components in lower levels. Three of the components, or Draco component directories, used by Milagro are the RNG, MC, and IMC packages. Tables 2 thru 4 list the component test problems for each of these components.

3.2. Future Tests

As stated in the previous section, additional component tests are required to gain more complete coverage of the Draco IMC components. Table 5 provides a list of classes in RNG and IMC that most urgently require component tests.

TABLE 4: Component tests in the Draco IMC package.

Problem	Description
tstOpacity	tests the Opacity_Builder, Opacity, and Mat_State classes; tests building, accessors, and interactions with the mesh
tstTally	tests the Tally class; tests tally accumulation and accessors
tstParticle	tests the Particle and Particle_Buffer classes; tests particle creation, communication, storage, equality, and accessors

TABLE 5: Classes in IMC and RNG that require testing or further testing.

Component	Class
RNG	`Rnd_Control`
IMC	`Particle` (expanded coverage)

4. Level 2: Test Problems

In this section we identify several key components of a calculation in Milagro. Several simple test problems are designed to exploit these key components. A python script runs the test suite, compares the test output to pre-existing benchmark outputs, and reports whether the test passed or failed. The python script can be run manually or within the general, overriding component/regression python script (the replacement for **dejagnu**). The test suite is useful for both regression testing and component testing at a high, compiled-code level.

4.1. Physical and Parametric Components

Considering an XYZ box as the system, we identify several components that together constitute the capabilities of Milagro. Several variants of the components are designed to give redundantly equivalent results, either exactly or statistically. For example, the results from an XYZ box should be invariant of the XYZ octant in which the box resides. The material definitions are no longer tested in Milagro because Milagro uses the Common Data Interface (CDI) [6, 7] to access opacities and equations-of-state.

1. Geometry
   - Cartesian
   - RZ
2. Translated Geometry
   - Each of the “octants”—8 for XYZ
   - origin at center of system
3. Mesh Definition
   - along a direction, 1 coarse cell with N fine cells
   - along a direction, N coarse cells, each with 1 fine cell
4. Material Definition
   - homogeneous
   - heterogeneous with like materials
   - heterogeneous
5. Frequency Treatment
   - gray
   - multigroup
6. Streaming
   - surface source (SS) straight through
   - SS with one opposing reflecting wall

- SS with all walls reflecting

7. Steady-State, Infinite Medium
   - all walls reflecting
   - one wall replaced by vacuum boundary condition and a SS at system temperature
   - two opposing walls replaced with vacuum boundary condition and SS at system temperature
8. Sources
   - Surface Source
     - six different faces
     - temperature, angular distribution
     - multiple (opposing only) faces
     - user-defined surface source cells
   - Material Volume Source
     - material and radiation decoupled
     - equilibrium diffusion in an infinite, homogeneous medium
   - Radiation Source
9. Isotropic Scattering
   - c=0
   - c=0.5
10. Data and Parameters
    - Absorption/Emission Coefficient/Opacity
      - analytic
      - analytic in tables
      - real data in tables
    - Specific Heat, c_v
      - analytic
      - analytic in tables
      - real data in tables
    - Fleck Time-Implicitness, $\alpha = 0, 1$
    - Buffer Size
      - $b = 1$
      - $b >> 1$
    - Density
      - “normal”
      - “large”
    - Random Number Seed
11. Parallelism [8]
    - full replication
    - full domain decomposition

4.2. **Test Problems**

The test problems are designed to test certain components or certain couplings of components. Unlike actual component testing, these regression tests test components from a higher, compiled-code level. Beyond component testing, the regression tests fulfill the traditional purpose of assuring that nothing unexpected has happened to the code.

4.2.1. *Orthogonal Structured Mesh.* The base test problem consists of a $4 \times 4 \times 4$ cm block with 4 cells in each direction. The base material has unit density, a specific heat of 0.1 jks/g/keV, and unit opacity/cross section. The computational parameters for this base test problem are a timestep of 0.01 shakes, 100 particles, buffer size of 1, and a random number seed of 9347593. The surface source angular distributions are cosine (which implies an isotropic intensity) in the infinite medium cases and normal in the streaming cases. The steady-state, infinite medium problems nominally start out at 1.0 keV with all six walls reflecting. The streaming problems have zero opacity, which implies no material interaction and no volume emission.

In July, 2000, we noticed an error in the infinite medium problems that had "T-cubed" specific heats, $c_v = c_{vo}T^3$: inf17 and inf19 (look ahead to Tables 6 and 11). The material energy and temperature monotonically increased in time to unrealistic levels. We traced the error to two things: the fact that our material data is time-explicit (evaluated at the beginning of the time step), and the fact that, given a stochastic temperature T that oscillates around an expected value, the cube, T^3, will be biased to a value larger than the expected value. To wit, 1.1^3 is farther away from 1.0 than is 0.9^3. At the end of a time step, the material temperature is updated by a quantity equal to the energy deposition divided by T^3. In these steady-state problems, the energy deposition in a time step should be distributed symmetrically about zero. Unfortunately, dividing by the T^3 from the beginning of the time step gives more weight to positive energy depositions. Thus, the material energy and temperature increases each cycle. Besides using time-implicit c_v values, using a shorter timestep is the best brute force fix. Increasing the particles helps reduce the magnitude of the bias, but does not eliminate it. We will not try to remedy this situation for these three infinite medium problems; they still serve their purpose for regression testing.

The infinite medium and streaming problem do not, unfortunately, isolate certain source-types of particles. For instance, they all have census particles. Three more test problems, which are part of the "start" series, are problems that totally consist of either surface source particles, volume emission particles, or census particles. These test problems perform restart-dumps, and the corresponding "restart" series of test problems picks them up from the first or second cycle and runs them to completion. The first set of problems, (re)start01-03, is run both serially and in parallel (Full Replication (Rep)). The second set, (re)start04-06, is a rehash of the first set and is run in parallel using a full domain decomposition (DD) topology. The next set of restart runs, restart07-09, pick up the corresponding restart-dumps from (re)start01-03 (Full Rep) and restart using Full DD. The next set of restart runs, restart10-12, pick up the restart-dumps from (re)start04-06 (Full DD) and restart using a Full Rep parallel topology. There are only three different physical problems in the start/restart set of calculations. The results for each physical problem should only differ by the number of cycles performed. Thus, problems 01, 04, 07, and 10 are all the same; as are 02, 05, 08, and 11; and as are 03, 06, 09, and 12.

In addition to the degenerate infinite medium, streaming, and restart (with isolated source-types) problems, we also run a few benchmark problems for a few cycles each. These problems have one-dimensional analytic solutions to which we compare more fully in the level 3 tests. Besides cursory regression testing, running a few cycles provides a check for catastrophic errors in these benchmark problems.

The regression test suite is categorized in the following list. Each item references a table that gives more details about the problem or deviations from its corresponding base problem.

- infinite medium, steady-state; see Table 6 for serial and Table 11 for parallel

TABLE 6: Serial, steady-state, infinite medium problems.

Problem name	Details, deviations from base test problem
inf01	Octant 1 (+x,+y,+z)
inf02	Octant 2 (–x,+y,+z)
inf03	Octant 3 (–x,–y,+z)
inf04	Octant 4 (+x,–y,+z)
inf05	Octant 5 (+x,+y,–z)
inf06	Octant 6 (–x,+y,–z)
inf07	Octant 7 (–x,–y,–z)
inf08	Octant 8 (+x,–y,–z)
inf09	Straddling the origin (±x,±y,±z)
inf10	inf09 with 2 coarse cells in each direction, 2 fine cells per coarse cell
inf11	inf10 with 2 materials, zonemap: 1 2 1 2 2 1 2 1
inf12	inf11 with 2 opposing SS(at system temperature) at low x and high x
inf13	inf11 with 2 opposing SS(at system temperature) at low y and high y
inf14	inf11 with 2 opposing SS(at system temperature) at low z and high z
inf15	inf09 with a different random number seed, 1234567
inf16	inf09 with explicit time differencing, α=0
inf17	inf09 with coefficient of $1/T^3$, specific heat of $0.1T^3$
inf18	inf09 with coefficient of 1.0, specific heat of 0.1
inf19	inf09 with coefficient of 1.0, specific heat of $0.1T^3$
inf20	inf09 with a large density, ρ=100 g/cc
inf21	inf12 with entire SS face defined by user (defined_surcells)
inf22	inf13 with entire SS face defined by user (defined_surcells)
inf23	inf14 with entire SS face defined by user (defined_surcells)
inf24	constant external material volume source (10Jks/cc/sh) in decoupled ($\sigma = 0$) one-cell infinite medium.
inf25	constant external material volume source (10Jks/cc/sh) in equilibrium ($\sigma = 100$) one-cell infinite medium.
inf26	inf18 except with $\kappa = 0.1 + 0.9/T$.
inf27	inf01 except with coefficients from an ipcress file.

- purely streaming, see Table 7 for serial and Table 11 for parallel
- isolated source-types and restart; see Table 8.
- verification problems run out to a few cycles, see Table 10

The infinite medium problems, inf01 through inf11, give exactly the same solution. Problems inf12 to inf15 and inf17 through inf19 are statistically equivalent to each other and to the first set, inf01 to inf11. Problem inf16 is different numerically because of its explicit time differencing and problem inf20 is a different problem. Problems inf24 and inf25 are entirely different problems since they are not steady-state problems and contain only one cell. Problem inf26 tests an opacity that goes as $a + b/T$. It is an original problem that exists merely for regression.

Problems str01 through str06 are the same, except that they have a different orientation. The same statement applies to the sets, str07 through str12 and str13 through str18.

Tables 8 and 9 describe "start" problems that contain only one of three species of source particles. These problems are then restarted in varying parallel topologies.

TABLE 7: Serial, streaming problems.

Problem name	Details, deviations from base test problem
str01	box around origin, vacuum b.c. on streaming axis, reflecting elsewhere, with SS on low x
str02	box around origin, vacuum b.c. on streaming axis, reflecting elsewhere, with SS on high x
str03	box around origin, vacuum b.c. on streaming axis, reflecting elsewhere, with SS on low y
str04	box around origin, vacuum b.c. on streaming axis, reflecting elsewhere, with SS on high y
str05	box around origin, vacuum b.c. on streaming axis, reflecting elsewhere, with SS on low z
str06	box around origin, vacuum b.c. on streaming axis, reflecting elsewhere, with SS on high z
str07	str01 with vacuum boundary on low x, reflecting elsewhere
str08	str02 with vacuum boundary on high x, reflecting elsewhere
str09	str03 with vacuum boundary on low y, reflecting elsewhere
str10	str04 with vacuum boundary on high y, reflecting elsewhere
str11	str05 with vacuum boundary on low z, reflecting elsewhere
str12	str06 with vacuum boundary on high z, reflecting elsewhere
str13	str01 with all boundaries reflecting
str14	str02 with all boundaries reflecting
str15	str03 with all boundaries reflecting
str16	str04 with all boundaries reflecting
str17	str05 with all boundaries reflecting
str18	str06 with all boundaries reflecting
str19 – str24	str01 – str06 with entire SS face defined by user (defined_surcells)
str25	str07 with SS on loz half of lox face
str26	str08 with SS on hiz half of hix face
str27	str09 with SS on center 2×2 block of loy face
str28	str10 with SS on outer ring of hiy face
str29	str11 with SS on diagonals of loz face
str30	str12 with SS defined on every other cell (checkerboard) of hiz face
str31	SS on center 2×2 block of each face

Given that Milagro is reproducible regardless of number of processors, the parallel regression problems give exactly the same results (within machine error) as their serial counterparts.

4.2.2. *RZ-Wedge Mesh.* The RZ-Wedge Mesh is the second Mesh Type upon which Milagro and its C++ classes have been templated. Where appropriate, we extend the Orthogonal Structured Mesh tests to the RZ-Wedge Mesh.

In the time between the development of the Orthogonal Structured Mesh regression tests and the RZ-Wedge regression tests, we discovered some machine-dependent rounding when there was supposed to be 3.5 particles in a cell. So now, since there are three species of particles in Milagro (volume emission, surface source, and census), the regression tests call for at least three particles per cell—or more if the energy is unbalanced between the three species—in order to get a nice, round number of particles per cell. This approach eliminates some machine-dependencies from testing the underlying physics. One consequence of this approach is that the comb is not tested in problems with all census particles (start03/06 family). For this reason, the start03/06 problems in the Orthogonal Structured mesh tests retain their "uneven" numbers of particles.

The base RZ-Wedge problem has $4 \times 4 = 16$ cells, 4 cm in each of the radial and axial dimensions, a wedge angle of 10 degrees, reflecting boundary conditions on all four sides, and material properties from the

TABLE 8: Benchmark "start" problems with particles from only one source-type. Each is restarted as a "restart" problem from the second, third, or fourth cycle. All problems are run in parallel; the first three in serial. Restarts can optionally change the parallel topology from Full Replication (Rep) to Full Domain Decomposition (DD) or vice versa.

Problem name	Description
start01/restart01	str01 except all surface source ($\Delta t = 0.1$, $\Delta x = 0.02$), full Rep
start02/restart02	inf16 except all volume emission ($\kappa = 10^4/T^3$, $\alpha = 0$), full Rep
start03/restart03	inf09/str13 except all census ($T_{mat} = \kappa = 0$, $\Delta t = 0.001$,$T_{rad} = 1.0$), full Rep
start04/restart04	start01/restart01, except full domain decomposition
start05/restart05	start02/restart02, except full domain decomposition
start06/restart06	start03/restart03, except full domain decomposition
restart07	picks up a restart01 dump (full Rep), restarts with full domain decomposition
restart08	picks up a restart02 dump (full Rep), restarts with full domain decomposition
restart09	picks up a restart03 dump (full Rep), restarts with full domain decomposition
restart10	picks up a restart04 dump (full DD), restarts with full replication
restart11	picks up a restart05 dump (full DD), restarts with full replication
restart12	picks up a restart06 dump (full DD), restarts with full replication

TABLE 9: A second set of benchmark "b_start" problems with particles from only one source-type. Each is restarted as a "b_restart" problem from the second, third, or fourth cycle.

Problem name	Description
b_start01	inf16, except $\kappa = 100 + 9900/T$
b_restart01	restarts start07 from cycle 2 (full rep)
b_restart02	restarts start07 from cycle 2 (full DD)

Orthogonal Structured Mesh tests. The base streaming test problem is the same geometry as the infinite medium test problems, is centered about the $z = 0$ axis, considers normally incident fluxes, and has all vacuum boundary conditions. The philosophical interpretation of these tests and comparisons between these tests are exactly the same as those of the Orthogonal Structured Mesh tests. Thus, we provide no further discussion. Table 12 shows the infinite medium test problems for the RZ-Wedge Mesh. Table 13 shows the streaming test problems for the RZ-Wedge Mesh. Table 14 shows the "start" and "restart" test problems for the RZ-Wedge Mesh. Table 15 lists the truncated versions of the full verification problems, which are essentially the same as for the orthogonal structured mesh. As with the orthogonal structure mesh case, the "fullrztp??" problems are meant to be executed less frequently than nightly. Instead, the "rztp??" problems, which are the same as the "fullrztp??" problems except only out to five cycles, are run nightly.

4.2.3. *Multigroup Frequency Treatment.* As of February 2002, the IMC and MC components in Draco and the Milagro code have been updated to use a multigroup frequency treatment in addition to a gray frequency treatment. Routines for sampling a Planckian and for sampling cumulative distribution function (for emission spectra) have been added and thoroughly component-tested.

The multigroup Milagro regression tests include all of the "start??/restart??" problems listed in Table 8 and those listed in Table 16. Most of the tests in Table 16 are inherently gray. These problems are run with the multigroup Milagro executable with a 3-group structure of frequency-independent opacities so as to replicate the gray tests. The frequency group bounds are 0.0, 0.2, 3.0, and 100.0 keV. All cases will use

TABLE 10: Serial benchmark problems run out a few cycles.

Problem name	Description
tp01	Marshak-2b [9] (Surface source impinging cold slab)
tp02	Marshak-1d [10] (Delta function source (in time and space))
tp03	Su/Olson non-equilibrium transport benchmark [11], $c = 0$
tp04	Su/Olson non-equilibrium transport benchmark [11], $c = 0.5$
tp05	Olson Wave [12]
tp06	Marshak-2a
tp07	Modified Marshak-2b (specific heat increased from 0.1 to 1.0)

TABLE 11: Parallel test problems, all on two processors.

Problem name	Description
p2_inf01	inf09, full replication
p2_inf02	inf09, full domain decomposition, buffer size = 1
p2_inf03	inf09, full domain decomposition, buffer size = 1000
p2_inf04	inf25, full replication
p2_str01	str17, full replication
p2_str02	str18, full replication
p2_str03	str17, full domain decomposition, buffer size = 1
p2_str04	str18, full domain decomposition, buffer size = 1
p2_str05	str31, full domain decomposition, buffer size = 1
p2_str06	str31, full replication, buffer size = 1

input analytic opacities, except for “mginf02” which utilizes an “ipcress” data file whose lower group bound is 1.0e-10 instead of zero.

The “mginf02” test utilizes an ipcress file called “tcube.ipcress”. It was constructed using tops (version topsn57) on the SGI machine theta with the following commands:

- source analy
- tlin 500 0.5 1.5
- g 1.0e-10 0.2 3.0 100.0
- sig0 1.0
- stem f
- ropow 0.0
- topow -3.0
- hnupow 0.0
- go
- end

The “go” command in tops prints out an ipcress file called “bliss,” which we rename “tcube.ipcress”. This data file is intended to replicate the analytic input for test “mginf01.” Given the lack of frequency dependence, the fine temperature grid, and the logarithmic interpolation, the results from “mginf02” did indeed match exactly the results from “mginf01.”

TABLE 12: Serial, steady-state, infinite medium problems for the RZWedge Mesh.

Problem name	Details, deviations from the base RZWedge test problem
rzinf01	+z regime
rzinf02	-z regime
rzinf03	straddle z=0
rzinf04	rzinf01 with 2 coarse cells in each direction, 2 fine cells per coarse cell
rzinf05	rzinf01 with 2 identical materials
rzinf06	rzinf05 with 2 opposing SS(at system temperature) at low z and high z
rzinf07	rzinf05 with SS(at system temperature) at high r
rzinf08	rzinf06, except straddling z=0
rzinf09	rzinf07, except straddling z=0
rzinf10	rzinf03, with coefficient of 1.0, specific heat of $0.1T^3$
rzinf11	rzinf08 with user-defined surface source cells
rzinf12	rzinf09 with user-defined surface source cells
rzinf13	constant external material volume source (10Jks/cc/sh) in decoupled ($\sigma = 0$) one-cell infinite medium.
rzinf14	constant external material volume source (10Jks/cc/sh) in equilibrium ($\sigma = 100$) one-cell infinite medium.
rzinf15	rzinf05, except with $\sigma = 0.1 + 0.9/T$.

TABLE 13: Serial, streaming problems for the RZWedge Mesh.

Problem name	Details, deviations from the base RZWedge test problem
rzstr01	surface source on high r, all vacuum boundaries
rzstr02	surface source on low z, all vacuum boundaries
rzstr03	surface source on high z, all vacuum boundaries
rzstr04	surface source on low z, reflecting on high z
rzstr05	surface source on high z, reflecting on low z
rzstr06	surface source on high r, all reflecting boundaries
rzstr07	surface source on low z, all reflecting boundaries
rzstr08	surface source on high z, all reflecting boundaries

The verification test problems and their correspondinly simpler regression test problems consist of one inherently gray problem and two non-grey problems. The mgtp01/mgfulltp01 problem is a Marshak-2A problem with the same aforementioned 3-group structure as above: 0.0 0.2 3.0 100.0.

The two problem sets, mgtp02/mgfulltp02 and mgtp03/mgfulltp03, compare Milagro's multigroup results to analytic multigroup results from Su and Olson's non-grey benchmarks [13]. Both problems have a picket fence opacity. Case B has one opacity 10 times the other, and Case B has one opacity 100 times the other. Theoretically, these analytic problems could be modeled with only two groups. However, since Milagro assumes an underlying Planckian distribution for its radiation sources and emission, the number of groups containing the alternating opacities must be large enough so that the Planckian shape between neighboring groups does not adversely weight one opacity more than the other. For Milagro's model of Case B, mgtp02/mgfulltp02, we used 64 groups, uniformly constructed over the range 0.0 to 100.0 keV. For Milagro's model of Case C, mgtp03/mgfulltp03, we used 128 groups, uniformly constructed over the range 0.0 to 102.4 keV. These numbers of groups were determined with a convergence study. We also found that to match Su and Olson's first early-time edit, we needed to refine the mesh by a factor of 8 and the timesteps by a factor of 100.

TABLE 14: RZ-Wedge Mesh benchmark "start" problems with particles from only one source-type.

Problem name	Description
rzstart01/rzrestart01	rzstr02 except all surface source ($\Delta t = 0.1$, $\Delta x = 0.02$)
rzstart02/rzrestart02	similar to inf16 except all volume emission ($\kappa = 10^4/T^3$, $\alpha = 0$)
rzstart03/rzrestart03	rzinf03/rzstr06 except all census ($T_{mat} = \kappa = 0$, $\Delta t = 0.001$,$T_{rad} = 1.0$)

TABLE 15: RZ-Wedge versions of the truncated benchmark problems—the full verification problem run out to five cycles.

Problem name	Description
rztp01	Marshak-2b [9] (Surface source impinging cold slab)
rztp02	Marshak-1d [10] (Delta function source (in time and space))
rztp03	Su/Olson non-equilibrium transport benchmark [11], $c = 0$
rztp04	Su/Olson non-equilibrium transport benchmark [11], $c = 0.5$
rztp05	Olson Wave [12]
rztp06	Marshak-2a
rztp07	Marshak-2b (c_v increased from 0.1 to 1.0)

Currently, the tally has not been upgraded to multigroup. When it is, we will add another test problem that is an infinite medium, steady-state problem with all reflecting surfaces except for one, which will be a surface source at the temperature of the medium. For a constant opacity, the escaping spectrum should be Planckian.

4.3. **Regression Test Scripts**

A python script, regress_milagro.py, runs the regression test. The serial option (python regress_milagro.py –serial) runs the problems in Tables 6, 7, and 10. The parallel option ("python regress_milagro.py –nprocs #", where # must be 2 unless "–version" is requested) runs the problems in Table 11. The serial calculations require about 5 minutes on an SGI Octane and the parallel calculations require about 0.5 minutes. The output goes to *problem_name*.test. The output is compared to *problem_name*.bench. Certain differences, such as number of particles or zeroth-order energies, indicate failure. Soft checks, such as energy checks or energy losses, are ensured to be only at machine error.

The benchmark files were verified by hand. Any discrepancies from previous executables from the past year

TABLE 16: Component and verification tests for Milagro with a multigroup frequency treatment.

Problem name	Description
mginf01	inf09
mginf02	inf09, except with an ipcress data file
mginf03	inf12
mginf04	inf25
mgstr01	str18
mgtp01/mgfulltp01	Marshak-2A
mgtp02/mgfulltp02	Su/Olson Non-Grey benchmark - Case B [13]
mgtp03/mgfulltp03	Su/Olson Non-Grey benchmark - Case C [13]

were verified to be caused by major changes in the code.

5. Level 3: Verification Problems

There exist a few one-dimensional problems with analytical solutions to which we compare Milagro. These are the same problems from Table 10, except that they are run out to several shakes. We run these problems in three-dimensional XYZ geometry, but with transverse directions made infinite with reflecting boundaries. These full test problems are listed in Table 17, where we also list the names for the RZ-Wedge Mesh tests.

TABLE 17: Benchmark test problems.

Problem name	Description
fulltp01/fullrztp01	Marshak-2b [9] (Surface source impinging cold slab)
fulltp02/fullrztp02	Marshak-1d [10] (Delta function source (in time and space))
fulltp03/fullrztp03	Su/Olson non-equilibrium transport benchmark [11], $c = 0$
fulltp04/fullrztp04	Su/Olson non-equilibrium transport benchmark [11], $c = 0.5$
fulltp05/fullrztp05	Olson Wave [12]
fulltp06/fullrztp06	Marshak-2a
fulltp07/fullrztp07	Modified Marshak-2b (c_v increased from 0.1 to 1.0)

The full Marshak 2B problem (fulltp01) has a density of 3 g/cc, an absorption coefficient of 100 cm^2-keV^3/g/T^3, a specific heat of 0.1 Jks/g/keV, and a cold temperature of 10^{-6} keV. The mesh thickness is 0.005 cm, the timestep is a constant 0.001 shakes, and 10,000 cycles were run using 10,000 particles per timestep.

The full Marshak 1D problem (fulltp02) begins from 0.1 shake with the analytic data as an initial condition. The cell thickness in the x-direction 0.0025 cm. The material has a density of 3 g/cc, and opacity of 1 cm^2-keV^3/g/T^3, a specific heat of 0.1 Jks/g/keV, and a cold temperature of 10^{-6} keV. The problem is run out to 10 shakes with a timestep of 0.01 sh and 10,000 particles.

The Su/Olson benchmarks (fulltp03 and fulltp04) are run out to a third of a shake with a timestep of $1/3 \times 10^{-4}$ sh. The radiation source existed from x=0 to x=0.5 cm for $1/3 \times 10^{-2}$ sh. The coefficient is 1.0 cm^2/g and the specific heat is 0.05488 Jks/cm^3/keV^4 T^3. The benchmark without scattering (fulltp03) uses a constant 10,000 particles, whereas the benchmark with 50% scattering begins with 5,000 particles and ramps up to 50,000 near the end of the calculation.

The Olson Wave (fulltp05) material has a density of 0.38214 g/cc, a coefficient of 2.61684 cm^2-keV^3/g/T^3, a cold temperature of 0.56234 keV, and a specific heat of 0.14361 Jks/g/keV. This problem is run out to 1/3 sh using a constant timestep of $1/3 \times 10^{-4}$ sh. The number of particles begins at 1000 and ramps up to 30,000 near the end of the problem.

The Marshak 2A problem (fulltp06) is similar to the Marshak 2B problem except that the absorption coefficient is order of magnitude smaller: 10 cm^2-keV^3/g/T^3. Other differences are that the timestep is a constant 0.0001 sh and the problem is run out to 0.1 shakes.

6. Running Milagro Regression Tests

All the regression test levels described in this note can be executed through the Draco and Milagro build systems. The description of the build system is described elsewhere [14]; however, a few notes about executing the regression tests are appropriate here.

To automatically execute regression tests through the build system for level 1 and 2 tests, one simply types

```
gmake check
```

in the target build directory for Draco and Milagro. This command will build all components, test them through the general python script (formally **dejagnu**), and install them in the appropriate places as set during configuration. For level 3 tests, which are only available in Milagro, one enters

```
gmake verify
```

in the Milagro build target directory.

The test can also be executed manually. Manual execution will not activate the general, overriding component/regression script (formally **dejagnu**) to log success and failure reports. To execute component tests manually the tests must first be built. For example, to execute the IMC component tests the following steps must be performed in the `draco/` target build directory:

```
gmake
cd draco/src/imc/test
gmake
./tstOpacity
./tstTally
./tstParticle
```

If this were a parallel (MPI) build, the `mpirun -np` *n* command would precede the executables.

References

[1] T. M. Evans and T. J. Urbatsch, "MILAGRO: A parallel Implicit Monte Carlo code for 3-d radiative transfer (U)," in *Proceedings of the Nuclear Explosives Code Development Conference*, (Las Vegas, NV), Oct. 1998. LA-UR-98–4722.

[2] T. Evans, "The Draco system for XTM transport code development," Research Note XTM-RN(U)-98–046, Los Alamos National Lab., 1998. LA-UR-98–5562.

[3] J. Lakos, *Large-Scale C++ Software Design*. Reading, MA: Addison-Wesley, Inc., 1996.

[4] R. Savoye, *The DejaGnu Testing Framework*. Free Software Foundation, 1.3, Jan. 1996.

[5] B. Meyer, *Object-Oriented Sofware Construction*. Upper Saddle River, NJ: Prentice Hall, second ed., 1997.

[6] K. G. Thompson, "Gandolf opacity package for draco," Technical Memo CCS-4:01-05(U), Los Alamos National Laboratory, May 2001.

[7] K. G. Thompson, "EOSPAC equation of state package for draco," Technical Memo CCS-4:01-17(U), Los Alamos National Laboratory, May 2001.

[8] T. J. Urbatsch and T. M. Evans, "Strategy for parallel Implicit Monte Carlo," Research Note XTM-RN(U)-98-018, Los Alamos National Laboratory, May 1998. LA-UR–98–2263.

[9] A. G. Petschek, R. E. Williamson, and J. K. Wooten, Jr., "The penetration of radiation with constant driving temperature," Technical Report LAMS–2421, Los Alamos Scientific Laboratory, July 1960.

[10] Y. B. Zel'dovich and Y. P. Raizer, *Physics of Shock Waves and High-Temperature Hydrodynamic Phenomena*. New York: Academic Press, 1966.

[11] B. Su and G. L. Olson, "An analytical benchmark for non-equilibrium radiative transfer in an isotrop-

ically scattering medium," *Annals of Nuclear Energy*, vol. 24, no. 13, pp. 1035–1055, 1997.

[12] G. L. Olson, L. H. Auer, and M. L. Hall, "Diffusion, P1, and other appoximate forms of radiation transport," in *Proceedings of the Nuclear Explosives Code Development Conference*, (Las Vegas, NV), Oct. 1998. LA-UR-98–5237.

[13] B. Su and G. L. Olson, "Non-grey benchmark results for two temperature non-equilibrium radiative transfer," *Journal of Quantitative Spectroscopy & Radiative Transfer*, vol. 62, pp. 279–302, 1999.

[14] T. Evans and R. Roberts, "The draco build system." In development, 1999.

Distribution:

Jim Morel, X–6, MS D409
Gordon Olson, X–6, MS D409
Grady Hughes, X–6, MS D409
William Krauser, X–2, MS B220
Joyce Guzik, X–2, MS B220
Michael Bernardin, X–2, MS B220
Robert Weaver, X–2, MS B220
Bernhard Wilde, X–2, MS B220
Don Shirk, X–8, MS F663
Eldon Linnebur, X–9, MS F663
Lauren Rauber, X–5, F664
Alexandra Heath, X–5, MS F663
Steve White, NIS–3, MS D440
Stephen Lee, NW–SC, MS F652
Mike Clover, X–11, MS F663
Johnny Collins, X–11, MS F663
Mike Gittings, X–11, MS F663
Gary Pfeufer, X–11, MS F663
John Romero, X–11, MS F663
Kim Simmons, X–11, MS F663

Todd Adams, X–6, MS D409
Ray Alcouffe, X–6, MS D409
Marv Alme, X–6, MS D409
Larry Auer, X–6, MS D409
Randy Baker, X–6, MS D409
Tom Evans, X–6, MS D409
Chris Gesh, X–6, MS D409
Mark Gray, X–6, MS D409
Mike Hall, X–6, MS D409
Henry Lichtenstein, X–6, MS D409
John McGhee, X–6, MS D409
Dimitri Mihalas, X–6, MS D409
Shawn Pautz, X–6, MS D409
Randy Roberts, X–6, MS D409
Scott Turner, X–6, MS D409
Todd Urbatsch, X–6, MS D409
Todd Wareing, X–6, MS D409
Jim Warsa, X–6, MS D409
X–6 Files, MS D409
XDO Files, MS B218

TJU:tju